```
//+------------------------------------------------------------------+
//|                                                  kase peak.mq4 |
//+------------------------------------------------------------------+
#property copyright "mladen"
#property link      "mladenfx@gmail.com"

#property indicator_separate_window
#property indicator_buffers  7
#property indicator_color1   DimGray
#property indicator_color2   DimGray
#property indicator_color3   Magenta
#property indicator_color4   DeepSkyBlue
#property indicator_color5   Magenta
#property indicator_color6   LimeGreen
#property indicator_color7   Red
#property indicator_width3   2
#property indicator_width4   2
#property indicator_width5   2
#property indicator_level1   0
#property indicator_levelcolor DimGray

//
//
//
//
//

extern double kcdDeviations           = 2.0;
extern int    kcdShortCycle           = 8;
extern int    kcdLongCycle            = 65;
extern double kcdSensitivity          = 40;
extern bool   allPeaksMode            = false;
extern bool   drawDivergences         = true;
extern bool   drawIndicatorTrendLines = true;
extern bool   drawPriceTrendLines     = true;
extern string drawLinesIdentificator  = "indicator1";
input bool           alertsOn         = true;              // Alerts true/false?
input bool           alertsOnCurrent  = false;             // Alerts open bar
true/false?
input bool           alertsMessage    = true;              // Alerts message
true/false?
input bool           alertsSound      = true;              // Alerts sound
true/false?
input bool           alertsPush       = false;             // Alerts push
notification true/false?
input bool           alertsEmail      = false;             // Alerts email
true/false?
input string         soundFile        = "alert2.wav";

//
//
//
//
//

double kchBuffer[];
double kcdBuffer[];
double kdeBuffer[];
double kcmBuffer[];
```

```
double kcpBuffer[];
double bullishDivergence[];
double bearishDivergence[],trend[];
double wrkBuffer[][6];
string shortName;
string labelNames;

//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
//
//
//
//
//

int init()
{
   IndicatorBuffers(8);
   SetIndexBuffer(0,kchBuffer); SetIndexStyle(0,DRAW_HISTOGRAM);
   SetIndexBuffer(1,kcdBuffer); SetIndexLabel(1,NULL);
   SetIndexBuffer(2,kdeBuffer);
   SetIndexBuffer(3,kcmBuffer);
   SetIndexBuffer(4,kcpBuffer); SetIndexStyle(4,DRAW_HISTOGRAM);
SetIndexLabel(4,"kpp");
   SetIndexBuffer(5,bullishDivergence); SetIndexStyle(5, DRAW_ARROW);
SetIndexArrow(5, 233);
   SetIndexBuffer(6,bearishDivergence); SetIndexStyle(6, DRAW_ARROW);
SetIndexArrow(6, 234);
   SetIndexBuffer(7,trend);

   labelNames = "Kase CD oscillator_DivergenceLine "+drawLinesIdentificator+":";
   shortName = "Kase CD oscillator div
("+DoubleToStr(kcdDeviations,2)+","+kcdShortCycle+","+kcdLongCycle+","+DoubleToStr(
kcdSensitivity,2)+")";
   IndicatorShortName(shortName);
   return(0);
}
int deinit()
{
   int length=StringLen(labelNames);
   for(int i=ObjectsTotal()-1; i>=0; i--)
   {
       string name = ObjectName(i);
       if(StringSubstr(name,0,length) == labelNames)  ObjectDelete(name);
   }
   return(0);
}




//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
//
//
//
//
```

```
//

#define ccLog 0
#define ccDev 1
#define x1    2
#define xs    3
#define xp    4
#define xpAbs 5

//
//
//
//
//

int start()
{
   int counted_bars=IndicatorCounted();
   int i,r,limit;

   if(counted_bars<0) return(-1);
   if(counted_bars>0) counted_bars--;
         limit = MathMin(Bars-counted_bars,Bars-kcdLongCycle);
         if (ArrayRange(wrkBuffer,0) != Bars) ArrayResize(wrkBuffer,Bars);

   //
   //
   //
   //
   //

   for(i=limit,r=Bars-i-1; i>=0; i--,r++)
   {
      wrkBuffer[r][x1]    = wrkBuffer[r-1][x1];
      wrkBuffer[r][xs]    = wrkBuffer[r-1][xs];
      wrkBuffer[r][ccLog] = MathLog(Close[i]/Close[i+1]);
      wrkBuffer[r][ccDev] = iDeviation(ccLog,9,r);

      //
      //
      //
      //
      //

      double avg = iSma(ccDev,30,r);
         if (avg>0)
         {
            double max1 = 0;
            double maxs = 0;
               for (int k=kcdShortCycle; k<kcdLongCycle; k++)
               {
                  max1 = MathMax(MathLog(High[i]/Low[i+k])/MathSqrt(k),max1);
                  maxs = MathMax(MathLog(High[i+k]/Low[i])/MathSqrt(k),maxs);
               }
            wrkBuffer[r][x1] = max1/avg;
            wrkBuffer[r][xs] = maxs/avg;
         }
         wrkBuffer[r][xp]    = kcdSensitivity*(iSma(x1,3,r)-iSma(xs,3,r));
         wrkBuffer[r][xpAbs] = MathAbs(wrkBuffer[r][xp]);
```

```
         //
         //
         //
         //
         //

         kcpBuffer[i+1] = EMPTY_VALUE;
         kcdBuffer[i]   = wrkBuffer[r][xp]-iSma(xp,9,r);
         kchBuffer[i]   = kcdBuffer[i];
            double tmpVal = iSma(xpAbs,50,r)
+kcdDeviations*(iDeviation(xpAbs,50,r));
            double maxVal = MathMax(90.0,tmpVal);
            double minVal = MathMin(90.0,tmpVal);

         if (kcdBuffer[i] > 0) { kdeBuffer[i] =  maxVal; kcmBuffer[i] =  minVal; }
         else                  { kdeBuffer[i] = -maxVal; kcmBuffer[i] = -minVal; }

      //
      //
      //
      //
      //

      if (!allPeaksMode)
      {
         if (kcdBuffer[i+1]>0 && kcdBuffer[i+1]>kcdBuffer[i] &&
kcdBuffer[i+1]>=kcdBuffer[i+2] && kcdBuffer[i+1]>= maxVal) kcpBuffer[i+1] =
kcdBuffer[i+1];
         if (kcdBuffer[i+1]<0 && kcdBuffer[i+1]<kcdBuffer[i] &&
kcdBuffer[i+1]<=kcdBuffer[i+2] && kcdBuffer[i+1]<=-maxVal) kcpBuffer[i+1] =
kcdBuffer[i+1];
      }
      else
      {
         if (kcdBuffer[i+1]>0 && kcdBuffer[i+1]>kcdBuffer[i] &&
kcdBuffer[i+1]>=kcdBuffer[i+2]) kcpBuffer[i+1] = kcdBuffer[i+1];
         if (kcdBuffer[i+1]<0 && kcdBuffer[i+1]<kcdBuffer[i] &&
kcdBuffer[i+1]<=kcdBuffer[i+2]) kcpBuffer[i+1] = kcdBuffer[i+1];
      }
      if (drawDivergences)
      {
         CatchBullishDivergence(i);
         CatchBearishDivergence(i);
      }
      trend[i]  = (i<Bars-1) ? (kcdBuffer[i]>0) ? 1 : (kcdBuffer[i]<0) ? -1 :
trend[i+1] : 0;
   }
   if (alertsOn)
   {
      int whichBar = (alertsOnCurrent) ? 0 : 1;
      if (trend[whichBar] != trend[whichBar+1])
      {
         if (trend[whichBar] == 1) doAlert(" crossing zero up");
         if (trend[whichBar] ==-1) doAlert(" crossing zero down");
      }
    }
   return(0);
}
```

```
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
//
//
//
//
//

double iDeviation(int forDim, int period, int pos)
{
   double dMA  = iSma(forDim,period,pos);
   double dSum = 0;
      for(int i=0; i<period; i++,pos--) dSum += (wrkBuffer[pos][forDim]-
dMA)*(wrkBuffer[pos][forDim]-dMA);
   return(MathSqrt(dSum/period));
}

//
//
//
//
//

double iSma(int forDim, int period, int pos)
{
   double sum = wrkBuffer[pos][forDim];
      for(int i=1; i<period; i++) sum += wrkBuffer[pos-i][forDim];
   return(sum/period);
}

//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
//
//
//
//
//

void CatchBullishDivergence(int shift)
{
   shift++;
         bullishDivergence[shift] = EMPTY_VALUE;
            ObjectDelete(labelNames+"l"+DoubleToStr(Time[shift],0));
            ObjectDelete(labelNames+"l"+"os" + DoubleToStr(Time[shift],0));

   if(!IsIndicatorLow(shift)) return;

   //
   //
   //
   //
   //

   int currentLow = shift;
   int lastLow = GetIndicatorLastLow(shift+1);
```

```
   //
   //
   //
   //
   //

   if(kcdBuffer[currentLow] > kcdBuffer[lastLow] && Low[currentLow] < Low[lastLow])
   {
      bullishDivergence[currentLow] = kcdBuffer[currentLow] - Point;
      if(drawPriceTrendLines == true)

DrawPriceTrendLine("l",Time[currentLow],Time[lastLow],Low[currentLow],Low[lastLow],
Green,STYLE_SOLID);
      if(drawIndicatorTrendLines == true)

DrawIndicatorTrendLine("l",Time[currentLow],Time[lastLow],kcdBuffer[currentLow],kcd
Buffer[lastLow],Green,STYLE_SOLID);
   }

   //
   //
   //
   //
   //

   if(kcdBuffer[currentLow] < kcdBuffer[lastLow] && Low[currentLow] > Low[lastLow])
   {
      bullishDivergence[currentLow] = kcdBuffer[currentLow] - Point;
      if(drawPriceTrendLines == true)

DrawPriceTrendLine("l",Time[currentLow],Time[lastLow],Low[currentLow],Low[lastLow],
Green, STYLE_DOT);
      if(drawIndicatorTrendLines == true)

DrawIndicatorTrendLine("l",Time[currentLow],Time[lastLow],kcdBuffer[currentLow],
kcdBuffer[lastLow], Green, STYLE_DOT);
   }
}


//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
//
//
//
//
//

void CatchBearishDivergence(int shift)
{
   shift++;
         bearishDivergence[shift] = EMPTY_VALUE;
            ObjectDelete(labelNames+"h"+DoubleToStr(Time[shift],0));
            ObjectDelete(labelNames+"h"+"os" + DoubleToStr(Time[shift],0));

   if(IsIndicatorPeak(shift) == false) return;
   int currentPeak = shift;
   int lastPeak = GetIndicatorLastPeak(shift+1);
```

```
   //
   //
   //
   //
   //

   if(kcdBuffer[currentPeak] < kcdBuffer[lastPeak] &&
High[currentPeak]>High[lastPeak])
   {
       bearishDivergence[currentPeak] = kcdBuffer[currentPeak] + Point;
       if(drawPriceTrendLines == true)

DrawPriceTrendLine("h",Time[currentPeak],Time[lastPeak],High[currentPeak],High[last
Peak],Red,STYLE_SOLID);
       if(drawIndicatorTrendLines == true)

DrawIndicatorTrendLine("h",Time[currentPeak],Time[lastPeak],kcdBuffer[currentPeak],
kcdBuffer[lastPeak],Red,STYLE_SOLID);
   }

   //
   //
   //
   //
   //

   if(kcdBuffer[currentPeak] > kcdBuffer[lastPeak] && High[currentPeak] <
High[lastPeak])
   {
      bearishDivergence[currentPeak] = kcdBuffer[currentPeak] + Point;
      if (drawPriceTrendLines == true)

DrawPriceTrendLine("h",Time[currentPeak],Time[lastPeak],High[currentPeak],High[last
Peak], Red, STYLE_DOT);
      if (drawIndicatorTrendLines == true)

DrawIndicatorTrendLine("h",Time[currentPeak],Time[lastPeak],kcdBuffer[currentPeak],
kcdBuffer[lastPeak], Red, STYLE_DOT);
   }
}

//
//
//

bool IsIndicatorPeak(int shift)
{
   if(kcpBuffer[shift] != EMPTY_VALUE && kcpBuffer[shift]>0)
        return(true);
   else return(false);
}
bool IsIndicatorLow(int shift)
{
   if(kcpBuffer[shift] != EMPTY_VALUE && kcpBuffer[shift]<0)
        return(true);
   else return(false);
}
int GetIndicatorLastPeak(int shift)
```

```
{
   for(int j=shift; j<Bars; j++) if(kcpBuffer[j] != EMPTY_VALUE && kcpBuffer[j]>0)
return(j);
   return(-1);
}
int GetIndicatorLastLow(int shift)
{
   for(int j=shift; j<Bars; j++) if(kcpBuffer[j] != EMPTY_VALUE && kcpBuffer[j]<0)
return(j);
   return(-1);
}

//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
//
//
//
//
//

void DrawPriceTrendLine(string first,datetime t1, datetime t2, double p1, double
p2, color lineColor, double style)
{
   string label = labelNames+first+"os"+DoubleToStr(t1,0);
   ObjectDelete(label);
      ObjectCreate(label, OBJ_TREND, 0, t1, p1, t2, p2, 0, 0);
         ObjectSet(label, OBJPROP_RAY, 0);
         ObjectSet(label, OBJPROP_COLOR, lineColor);
         ObjectSet(label, OBJPROP_STYLE, style);
}
void DrawIndicatorTrendLine(string first,datetime t1, datetime t2, double p1,
double p2, color lineColor, double style)
{
   int indicatorWindow = WindowFind(shortName);
   if (indicatorWindow < 0) return;

   string label = labelNames+first+DoubleToStr(t1,0);
   ObjectDelete(label);
      ObjectCreate(label, OBJ_TREND, indicatorWindow, t1, p1, t2, p2, 0, 0);
         ObjectSet(label, OBJPROP_RAY, 0);
         ObjectSet(label, OBJPROP_COLOR, lineColor);
         ObjectSet(label, OBJPROP_STYLE, style);
}

//
//
//

string sTfTable[] = {"M1","M5","M15","M30","H1","H4","D1","W1","MN"};
int    iTfTable[] = {1,5,15,30,60,240,1440,10080,43200};

string timeFrameToString(int tf)
{
   for (int i=ArraySize(iTfTable)-1; i>=0; i--)
         if (tf==iTfTable[i]) return(sTfTable[i]);
                              return("");
}
```

```
//-------------------------------------------------------------------
//   Alerts function
//-------------------------------------------------------------------

void doAlert(string doWhat)
{
   static string   previousAlert="nothing";
   static datetime previousTime;
   string message;

      if (previousAlert != doWhat || previousTime != Time[0]) {
          previousAlert  = doWhat;
          previousTime   = Time[0];

          //
          //
          //
          //
          //

          message = timeFrameToString(_Period)+" "+_Symbol+" at
"+TimeToStr(TimeLocal(),TIME_SECONDS)+" Kase CD "+doWhat;
             if (alertsMessage) Alert(message);
             if (alertsPush)    SendNotification(message);
             if (alertsEmail)   SendMail(_Symbol+" Kase CD ",message);
             if (alertsSound)   PlaySound(soundFile);
      }
}
```